Ricostruzione tridimensionale della struttura della cromatina da dati tipo Chromosome Conformation Capture

Progetto Bandiera InterOmics – WP1 CNR-ISTI

Claudia Caudai

Istituto di Scienza e Tecnologie dell'Informazione - CNR
Via G. Moruzzi, 1, I-56124 PISA, Italy
claudia.caudai@isti.cnr.it

Sommario Questo documento riporta l'analisi del problema, e un primo approccio alla soluzione, riguardo al programma di ricerca del WP1-ISTI InterOmics sull'analisi della struttura tridimensionale della fibra di cromatina all'interno del nucleo cellulare. Da un'analisi critica delle tecniche recentemente proposte in letteratura sulla base delle vastissime possibilità offerte dagli esperimenti Hi-C, è emerso che l'affidabilità e la significatività biologica dei risultati già ottenuti possono essere migliorate introducendo opportuna conoscenza a priori nella soluzione del problema. Quanto contenuto in questo rapporto costituisce la nostra prima risposta a queste esigenze, preliminare alla validazione definitiva delle procedure sviluppate, all'applicazione a dati significativi, e all'analisi dei risultati. Il nostro approccio parte da una considerazione sulla significatività dei dati iniziali che ci ha portato a formulare un nuovo tipo di criterio per il fit tra le configurazioni ricostruite e gli esperimenti. Il secondo aspetto del nostro intervento consiste nell'introduzione di conoscenza a priori sotto forma di un nuovo modello di soluzione su cui si possono forzare vincoli di tipo geometrico. Infine, dalla considerazione dell'esistenza di domini genomici che interagiscono poco tra di loro, viene anche fatta notare la possibilità di un approccio multiscala, che operi le ricostruzioni tridimensionali indipendentemente su diversi frammenti che possono poi essere sfruttati per ricostruire, ricorsivamente, tratti di catena più lunghi.

1 Introduzione

L'organizzazione spaziale del DNA all'interno della cellula è un problema molto studiato in biologia e bioinformatica. Capire la struttura tridimensionale del filamento di cromatina in cui è organizzato il DNA è molto importante per capire la struttura dei cromosomi, le loro interazioni, e la loro funzionalità. Grazie alla disponibilità delle moderne tecniche di sequenziamento genomico (Next Generation Sequencing), sono stati recentemente sviluppati metodi sperimentali, derivati dalle tecniche di Chromosome Conformation Capture (3C), che permettono di rilevare con un'alta risoluzione spaziale i contatti tra diversi segmenti

della cromatina nelle configurazioni che essa assume. Uno di essi, chiamato Hi-C [1], produce enormi quantità di dati, a livello dell'intero genoma, sulla cui base sono stati proposti molti metodi per ricostruire la conformazione tridimensionale della cromatina.

Da una popolazione di molti milioni di cellule, attraverso un'elaborata procedura di cross-linking, frammentazione, purificazione e sequenziamento, con il metodo Hi-C si riesce a contare quante volte i componenti di ogni possibile coppia di frammenti di DNA si trovano a contatto tra loro nelle varie configurazioni che la cromatina assume all'interno della popolazione esaminata. Applicato a un cromosoma umano trattato utilizzando l'enzima HindIII, Hi-C identifica contatti tra porzioni di DNA lunghe in media 4 kilobasi (kb, di qui in avanti). Questi dati sono organizzati in una *matrice dei contatti*, i cui elementi sono i numeri di contatti (eventualmente normalizzati) di ogni porzione di DNA tagliata dall'enzima con tutte le altre. Questa matrice sarà ovviamente sparsa, perché molte tra le possibili coppie di frammenti non si troveranno a contatto in nessuna delle configurazioni di cromatina. Per motivi legati alla trattabilità del problema di ricostruzione, e anche alla ripetibilità dell'esperimento, si tende a rendere non sparsa la matrice raggruppando insieme blocchi di elementi contigui e sommandone i valori (*binning*). Così facendo, si eliminano molti zeri ma si perde in risoluzione rispetto alle originarie 4 kb. Indipendentemente dalla risoluzione, i dati contenuti nella matrice dei contatti sono affetti da svariati tipi di errore, le cui entità possono essere ridotte [2] o tenute in conto dal modello su cui si basano gli algoritmi di ricostruzione tridimensionale [3].

I metodi proposti in letteratura per la ricostruzione 3D della cromatina da dati di contatto sono prevalentemente di carattere bayesiano. L'approccio tuttora più comune perturba il modello della cromatina, introducendo pochi (o nulli) vincoli fisici o geometrici, per raggiungere configurazioni il più possibile in accordo con i dati della matrice dei contatti [2, 4]. Questo tipo di approccio presenta quanto meno due ordini di problemi. Il primo è relativo all'accordo tra i dati e la struttura ricostruita: i dati provengono da un'intera popolazione di cellule, in cui la cromatina può trovarsi in configurazioni diverse tra loro, mentre la ricostruzione tratta una singola struttura, su cui non possono essere certo conteggiati i contatti. L'accordo con i dati non viene dunque verificato direttamente sulla matrice data: fidando sulla considerazione intuitiva che la rilevazione di un alto numero di contatti tra due frammenti corrisponde a una breve distanza tra loro in molte delle cellule analizzate, la matrice dei contatti viene trasformata in una matrice delle distanze per mezzo di formule deterministiche e semi-empiriche. Un secondo ordine di problemi riguarda la mancata introduzione di vincoli: nei risultati della procedura di ricostruzione, la coerenza chimico-fisica della struttura della cromatina ne risulta sacrificata.

Per citare un approccio specifico, in [4], la matrice dei contatti è trasformata ricorrendo a una relazione quadratica inversa tra il numero dei contatti e la distanza. Almeno nel caso di coppie con contatti frequenti, questo modo di procedere rispecchia una situazione fisica probabilmente giusta (molti contatti=breve distanza), ma nel caso di coppie che si toccano poco frequentemente

è troppo azzardato presumere che i relativi frammenti siano effettivamente lontani nello spazio, anche perché lo spazio nel nucleo della cellula è ristretto, e la maggior parte dei contatti sono funzionali e non aleatori. Un ulteriore aspetto problematico, sinora non toccato in letteratura, riguarda la coerenza geometrica di un sistema di distanze comunque generato a partire da dati di contatto. Partiamo da un elenco di questioni fondamentali nella critica degli attuali approcci. A ognuna delle domande, affianchiamo una possibile risposta preliminare, che tenteremo di completare o precisare sulla base dei futuri esperimenti condotti con i nuovi metodi che intendiamo sviluppare:

1. **Ha senso integrare dati da un'intera popolazione in un'unica matrice dei contatti?** Sì, se le cellule prese in esame sono simili, perché si presume che abbiano gli stessi contatti funzionali.
2. **Ha senso binnare la matrice?** Dal punto di vista biologico, ridurre la risoluzione dei dati impedisce di tenere traccia dei contatti puntuali più specifici, che rivestono spesso significati funzionali importanti. Se una binnatura è comunque necessaria per rendere più facilmente gestibile la matrice, è preferibile, se possibile, utilizzare una binnatura moderata (per esempio 50-100 kb, tenendo presente che l'intero genoma contiene $3 \cdot 10^9$ basi, suddivise in comosomi di 20-150 Mb).
3. **Ha senso trasformare la matrice dei contatti per ottenere una matrice delle distanze geometriche?** Per rispondere a questa domanda occorre verificare il grado di coerenza della trasformazione. A priori, non sappiamo dire se esiste una trasformazione che permetta di tradurre in modo coerente un sistema di contatti (eventi binari di natura funzionale ed aleatoria) con un sistema geometrico di distanze euclidee nello spazio tridimensionale.

Dopo uno studio di stato dell'arte, lo scopo del WP1 ISTI riguardo al problema qui descritto è di sviluppare una o più tecniche di ricostruzione che partano da queste domande, e presumibilmente da altre che verranno nel corso dello studio, per superare le difficoltà teoriche e pratiche ad esso legate, e precisare il significato dei risultati ottenuti, sia dal punto di vista metodologico e algoritmico sia, possibilmente, dal punto di vista biologico.

Questo rapporto è organizzato come segue. Al Paragrafo 2, sono riportati i risultati della nostra analisi sulla consistenza geometrica delle matrici delle distanze ricavate a partire dalla matrice dei contatti. Ai Paragrafi 3 e 4, spieghiamo il nostro approccio al problema, sviluppato sulla base delle incongruenze riscontrate negli approcci attuali, mentre ai Paragrafi 5 e 6 descriviamo l'algoritmo in dettaglio e facciamo alcune considerazioni sulle sue caratteristiche. Infine, al Paragrafo 7 presentiamo i primi risultati ottenuti in fase di validazione del codice. L'appendice riporta l'ultima versione del codice Maple utilizzato per condurre gli esperimenti.

2 Coerenza della matrice delle distanze

Data la matrice dei contatti, si desidera ricavare da essa informazioni che possano far luce sulla topologia della cromatina. È importante riflettere sul fatto che stiamo cercando di estrarre informazioni topologiche (quindi misure, distanze) da dati che si riferiscono a contatti, quindi eventi a carattere discreto (somma di eventi dicotomici). La matrice dei contatti specifica, per ogni singolo punto della cromatina, il numero di volte in cui esso si è trovato a contatto con altri punti specifici. In modo intuitivo, si può poi assumere che due regioni genomiche frequentemente a contatto occupino anche regioni spazialmente vicine, indipendentemente dalla loro reciproca distanza genomica. Da tale riflessione hanno avuto origine alcuni percorsi di ricerca che hanno sondato metodi per trovare modelli plausibili della cromatina a partire dalla matrice dei contatti [3–5]. Il processo di estrazione di dati con valenza topologica dalla matrice dei contatti richiede molta attenzione. Una volta ottenuta una chiave di trasformazione adeguata, la ricostruzione topologica della cromatina può essere effettuata con differenti metodi, di carattere energetico, probabilistico, inferenziale, bayesiano, geometrico. Ancor prima di andare ad analizzare il metodo di ricostruzione della conformazione tridimensionale a partire dalle distanze tra alcuni punti specifici, è importante creare un insieme di distanze che rappresentino un sistema coerente geometricamente. Alcune osservazioni preliminari a livello geometrico:

1. La cromatina è racchiusa nel nucleo della cellula, pertanto le massime distanze possibili tra i suoi frammenti non possono superare il diametro del nucleo cellulare (tipicamente, tra 5 e 10 μm).
2. Un punto nello spazio è fissato quando sono individuate le sue distanze da altri tre punti fissi. La sua posizione è fissata a meno di una riflessione rispetto al piano che contiene i tre punti. È molto difficile che invertendo gli elementi di una matrice di frequenze di contatti si ottenga un sistema di distanze coerente con una configurazione tridimensionale euclidea.
3. Occorre verificare la compatibilità della matrice delle distanze con un sistema di distanze effettivamente riferito a un insieme di n punti nello spazio euclideo.

Occorre fare almeno delle verifiche preliminari per capire se il sistema di distanze rispetta le condizioni di costruibilità di base: si parla di diseguaglianze triangolari, e di diseguaglianze analoghe per poligoni di più lati. In pratica, date le distanze reciproche tra un numero qualunque di punti consecutivi in una catena, la distanza del primo dall'ultimo non può superare la somma delle distanze tra i punti intermedi. La non-violazione di tali diseguaglianze è una condizione necessaria ma non sufficiente per la coerenza geometrica. Se tali condizioni vengono violate in modo molto ingente, il sistema di distanze non potrà essere utilizzato come target per il raggiungimento di conformazioni geometriche tridimensionali della cromatina, perché si otterrebbero incoerenze importanti dal punto di vista strutturale. Al contrario, il fatto che tali diseguaglianze non vengano violate, o che vengano violate in modo lieve, non assicura la coerenza del

sistema, perché le distanze potrebbero essere eccessivamente ravvicinate, e questo non avrebbe ugualmente coerenza fisica con le possibili conformazioni della cromatina (per esempio, se tutti i punti fossero sovrapposti nell'origine, le condizioni di costruibilità non sarebbero violate ma il sistema non sarebbe ugualmente plausibile). Per indagare il problema, ho realizzato un programma in Maple che calcola il numero di infrazioni (ed il loro peso) per varie leggi di trasformazione usate per invertire differenti insiemi di dati, e ho valutato quali di queste leggi creano minori infrazioni di costruibilità nel processo di trasformazione dei dati da frequenze di contatto a distanze. Tale processo soffre di una incoerenza di fondo, fondamentalmente perché la frequenza dei contatti non ha una corrispondenza esclusivamente geometrica, ma può dipendere anche da altri fattori, quali la presenza di barriere topologiche, alcune condizioni energetiche, eventi aleatori. L'inversione ha senso, e rispecchia una effettiva situazione fisica, nel caso di punti con contatti frequenti, ma nel caso di punti che si toccano poco frequentemente è troppo azzardato presumere che essi siano lontani nello spazio, anche perché lo spazio nella cellula è ristretto e la cromatina subisce molti ripiegamenti su se stessa per riuscire ad entrare in tale spazio. Tramite il programma realizzato, ho confrontato il numero e il peso delle infrazioni utilizzando due insiemi di dati differenti:

1. I dati del genoma umano forniti dall'esperimento Hi-C realizzato da Lieberman-Aiden [1] (GM06690 raw data, observed, 1Mb binned);
2. I dati forniti da Yaffe e Tanay [2], ripuliti dai bias secondo l'algoritmo da loro proposto e presenti sulla loro pagina web[1] (GSE18199 expected data, 1Mb binned).

Per visualizzare le differenze tra questi due insiemi di dati, mostriamo in Figura 1 le relative matrici di contatto per il solo cromosoma 18. La scala di colore è logaritmica per rendere leggibili dati a dinamica elevata. La differenza tra questi due insiemi di dati, relativi allo stesso tratto di genoma, è più che evidente. A parte la variabilità sperimentale, una possibile ragione di questa differenza è la procedura di ripulitura dai bias che hanno subito i dati presentati in [2]. Per ognuno di questi due insiemi, ho considerato tutti i 23 cromosomi, e ho confrontato il numero di violazioni e la loro entità per 13 leggi differenti (x è il numero di contatti e d la distanza euclidea):

$$x \to d = \frac{1}{x^3} \tag{1}$$

$$x \to d = \frac{1}{x^2} \tag{2}$$

$$x \to d = \frac{1}{x} \tag{3}$$

$$x \to d = \frac{1}{\sqrt{x}} \tag{4}$$

$$x \to d = \frac{1}{\sqrt[3]{x}} \tag{5}$$

[1] http : //compgenomics.weizmann.ac.il/tanay/?page_id = 283

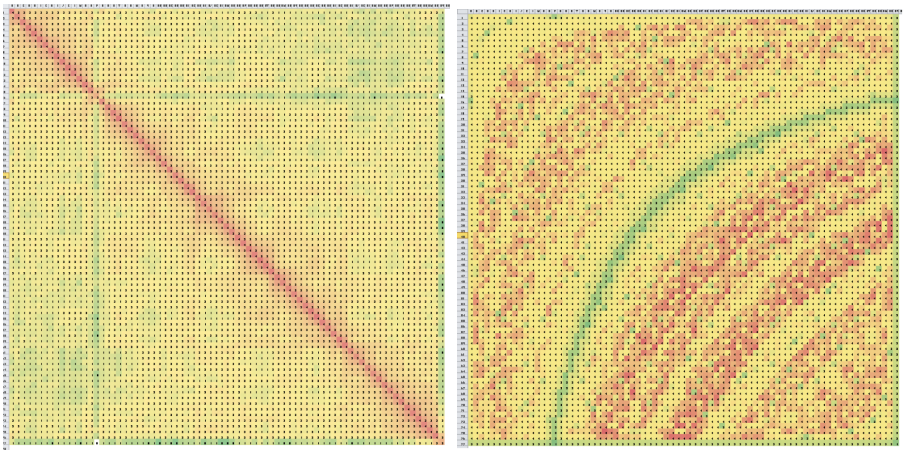

Figura1. Matrici di contatto per il cromosoma 18, da dati GM06690 [1] (a sinistra) e GSE18199 da [2] (a destra). Visualizzazione sotto forma di *heatmap*, con scala di colore logaritmica (verde=minimo, rosso=massimo).

$$x \to d = \frac{1}{\sqrt[4]{x}} \tag{6}$$

$$x \to d = \frac{1}{\sqrt[5]{x}} \tag{7}$$

$$\begin{array}{r} x \to d = \log^2(x), \text{per i dati in probabilità di contatto (YT)} \\ x \to d = \frac{1}{\log^2(x)}, \text{per i dati in numero di contatti (LA)} \end{array} \tag{8}$$

$$\begin{array}{r} x \to d = \log(x), \text{per dati YT} \\ x \to d = \frac{1}{\log(x)}, \text{per dati LA} \end{array} \tag{9}$$

$$\begin{array}{r} x \to d = \sqrt{\log(x)}, \text{per dati YT} \\ x \to d = \frac{1}{\sqrt{\log(x)}}, \text{per dati LA} \end{array} \tag{10}$$

$$\begin{array}{r} x \to d = \sqrt[3]{\log(x)}, \text{per dati YT} \\ x \to d = \frac{1}{\sqrt[3]{\log(x)}}, \text{per dati LA} \end{array} \tag{11}$$

$$\begin{array}{r} x \to d = \sqrt[4]{\log(x)}, \text{per dati YT} \\ x \to d = \frac{1}{\sqrt[4]{\log(x)}}, \text{per dati LA} \end{array} \tag{12}$$

$$\begin{array}{r} x \to d = \sqrt[5]{\log(x)}, \text{per dati YT} \\ x \to d = \frac{1}{\sqrt[5]{\log(x)}}, \text{per dati LA} \end{array} \tag{13}$$

I due grafici in Figura 2 rappresentano, per tutte le leggi considerate, il numero di violazioni delle condizioni di costruibilità geometrica che emergono dalle diverse matrici delle distanze derivanti dai due insiemi di dati. Analogamente, in Figura 3, sono riportate le entità totali delle violazioni, per ogni cromosoma e per ogni

legge considerata. Per ogni possibile terna di loci emergente dalla matrice dei contatti, si calcola se e di quanto (in percentuale) le distanze date violano la disuguaglianza triangolare; la somma di tutti i valori così ottenuti viene poi divisa per il numero totale delle terne considerate. Come si nota dalla figura, almeno per alcuni cromosomi e per le prime due leggi, la compatibilità tra il sistema di distanze dato e una struttura tridimensionale nello spazio euclideo è davvero bassa.

Dagli esperimenti condotti, è emerso che Il numero di violazioni inizia drasticamente a diminuire applicando la legge $1/\sqrt{x}$. Si ricava che le leggi con la radice al denominatore (insieme alle leggi che contengono il logaritmo, che hanno a loro volta un comportamento "smussante") sono più adatte per una trasformazione da contatti a distanze. Si osserva che, per entrambi gli insiemi di dati, esiste un trend di proporzionalità tra il valore dell'esponente al denominatore (per le leggi $1/x^n$) e il numero (e l'entità) delle violazioni. Le leggi meno adatte, e con un maggior numero di violazioni, sono $1/x^n$ con $n \geq 3$. Sembra che la legge $1/\sqrt{x}$ sia più adatta delle successive (con esponenti radicali più alti) perché più l'esponente della radice aumenta, più il radicando tenderà ad avvicinarsi a 1, quindi l'omogeneizzazione dei valori delle distanze potrebbe risultare eccessiva in casi come radice quarta o radice quinta. Lo stesso ragionamento vale per le funzioni che contengono il logaritmo. Il fatto che si abbiano risultati simili per insiemi di dati così diversi indica che questo tipo di verifica valuta effettivamente l'ammissibilità della legge di conversione esaminata e non qualche particolarità dell'insieme di dati utilizzato.

3 Calcolo diretto della configurazione in base alla matrice dei contatti

In base alle riflessioni sopra riportate, si è proceduto affrontando il problema della ricostruzione tridimensionale della cromatina direttamente dalla matrice dei contatti, senza trasformarla in matrice delle distanze, e con un approccio diverso rispetto a quelli fin qui adottati [5]. Questi partono dalla modellazione di un insieme di frammenti senza vincoli di alcun tipo, la cui distribuzione spaziale viene perturbata utilizzando un algoritmo Monte Carlo, con una funzione obiettivo derivata dalla trasformazione della matrice dei contatti in matrice delle distanze euclidee. Al contrario, abbiamo scelto di partire da un modello della cromatina che tiene conto di vincoli strutturali ben precisi: ogni frammento è una perla (impenetrabile) in una collana ordinata [6] dotata di una elasticità, un'allungabilità massima, e una massima curvatura ammessa. Tali vincoli sono messi in corrispondenza con caratteristiche biologiche, e costringono il modello a rispettare dimensioni e proporzioni reali della cromatina. Tale sistema fisico viene perturbato tramite un algoritmo Monte Carlo, che agisce sui tre parametri identificativi di ogni punto. A tale proposito, abbiamo scelto di non usare un sistema di coordinate cartesiane, ma un sistema quaternionico. Ogni punto nella catena è individuato da un angolo planare, un angolo diedrale, e dalla lunghezza del segmento che lo unisce al punto precedente. Questo segmento giace sull'asse

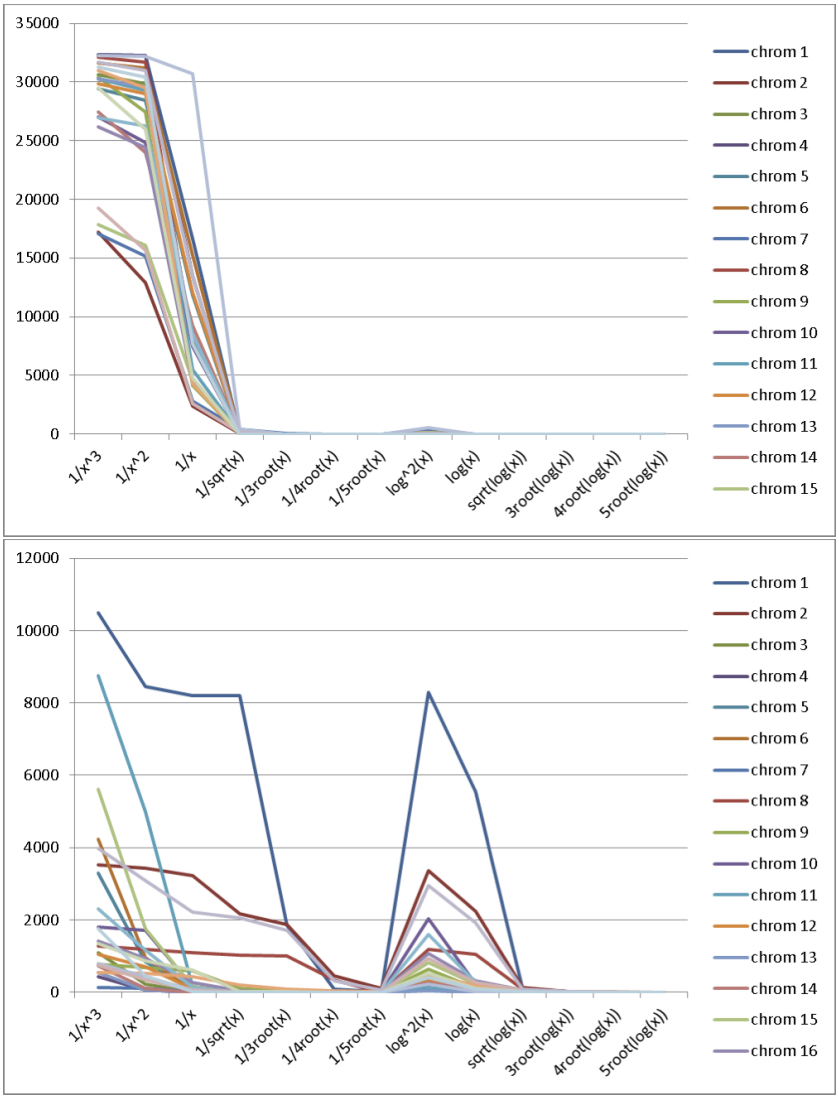

Figura2. Numero di violazioni delle condizioni di costruibilità per ogni cromosoma e per ognuna delle (1)–(13). In alto: dati GM06690 da [1]. In basso: dati GSE18199 da [2].

di rotazione del quaternione relativo all'angolo diedrale. La funzione obiettivo sfruttata per costruire l'algoritmo (detta anche funzione energia) si basa sulla matrice dei contatti, come detto, non tradotta in termini di distanze. Volendo individuare conformazioni plausibili sulla base della matrice dei contatti, abbiamo ritenuto lecito cercare le coppie di frammenti più frequentemente a contatto tra

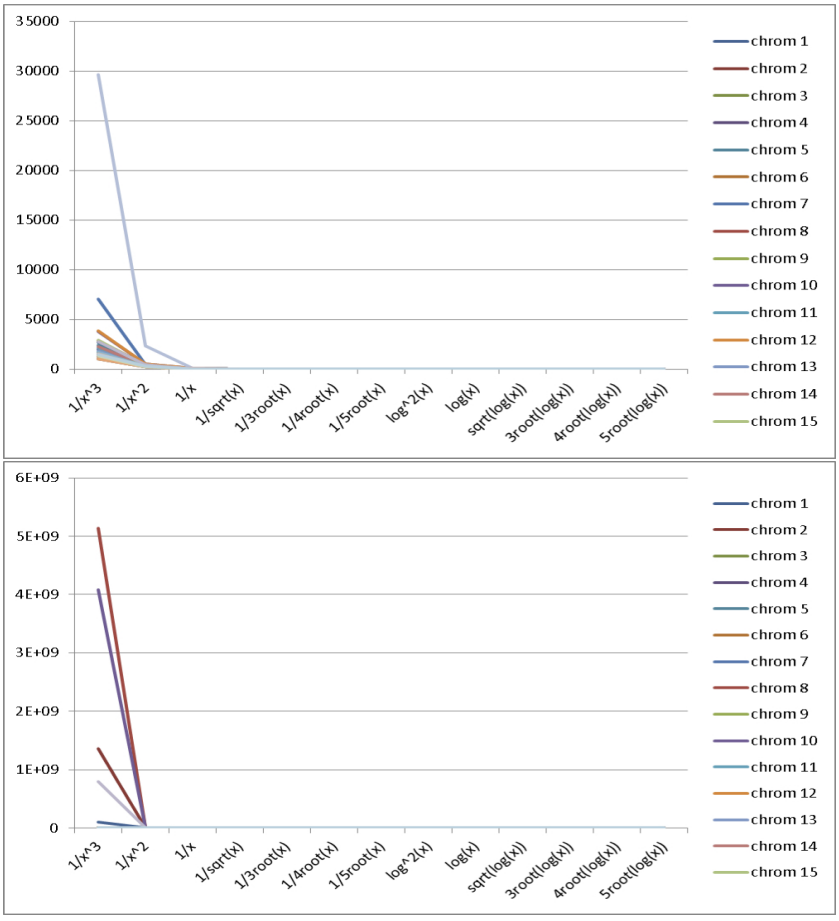

Figura3. Entità delle violazioni delle condizioni di costruibilità per ogni cromosoma e per ognuna delle (1)–(13). In alto: dati GM06690 da [1]. In basso: dati GSE18199 da [2].

loro, presumendo che esse caratterizzino configurazioni comuni a molte situazioni di equilibrio reali della cromatina, e che per questo motivo possano coesistere simultaneamente in un'unica configurazione plausibile. Il programma realizzato comprende un apposito filtro che seleziona solamente gli elementi con frequenza di contatto più alta, in modo da stabilire come obiettivo la vicinanza geometrica dei frammenti appartenenti alle corrispondenti coppie. Tendenzialmente, vogliamo che tali punti si trovino vicini tra loro nella configurazione finale, mentre gli altri elementi della catena, non vincolati dalla funzione obiettivo ma solo dai vincoli fisici e topologici stabiliti per il modello *a priori*, mantengono maggiore libertà di movimento. In questo modo, il sistema mantiene dei gradi di libertà e

permette di trovare diverse configurazioni con punti di contatto comuni, ma che possono subire ripiegamenti diversi. Se si riesce a stabilire che tali configurazioni diverse sono tutte presenti nella popolazione oggetto dell'esperimento Hi-C, esse potrebbero rappresentare diversi stati funzionali del tratto di genoma studiato o diverse fasi della dinamica della cromatina.

4 Approccio multiscala

Potendo sfruttare i dati direttamente in termini di numero di contatti, la presenza di zeri nella matrice non è un ostacolo alla ricostruzione della struttura. Questo approccio ci permette dunque di evitare eccessive (e arbitrarie) binnature della matrice dei contatti, con conseguente perdita di risoluzione.

Lo studio delle funzionalità dei domini genomici dispone di una ulteriore possibilità: quella di analizzare separatamente tratti diversi del genoma, magari caratterizzati da specifiche funzioni più o meno note, e quindi rimetterli insieme a una risoluzione più bassa per ottenere la configurazione globale della cromatina al livello del singolo cromosoma o addirittura dell'intero genoma. In pratica, si propone una procedura di ricostruzione ricorsiva: 1) si identificano i tratti significativi e la risoluzione alla quale si vuole ricostruirne la struttura; 2) si opera sulla matrice dei contatti originale per farle assumere la risoluzione voluta in corrispondenza di ogni tratto; 3) si opera in parallelo la ricostruzione delle strutture dei singoli tratti; 4) si raggruppano insiemi significativi dei tratti ricostruiti in modo da farli corrispondere a una nuova risoluzione della matrice dei contatti; 5) si ripete la procedura ricorsivamente dal passo 2) fino ad arrivare alla minima risoluzione voluta.

Nelle prime prove qui riportate, questo schema è stato seguito per due scale distinte, senza però farle corrispondere a partizioni note del cromosoma considerato: una prima ricostruzione è stata effettuata su dieci tratti composti ognuno da dieci frammenti lunghi 100 kb; una volta ottenute le strutture di questi tratti, ognuno di essi è stato considerato come un singolo frammento lungo 1 Mb e, tenendone fissa la configurazione, si è ricostruita la struttura 3D del tratto di cromatina considerato alla risoluzione di 1 Mb, sulla base della relativa matrice dei contatti.

5 Struttura del codice di calcolo

Il programma è scritto in Maple (Versione 17) e si divide in 4 parti, descritte nei successivi sottoparagrafi: lettura dei dati iniziali, prima procedura (bin 100 kb), seconda procedura (bin 1 Mb), ricostruzione dell'intera catena ad alta risoluzione, e visualizzazione dei risultati.

5.1 Lettura dei dati iniziali

Il programma legge due matrici dei contatti derivanti dallo stesso esperimento Hi-C e telative a uno stesso cromosoma, alle due diverse risoluzioni di 100 kb e 1 Mb, rispettivamente.

5.2 Prima procedura

La prima procedura gestisce il movimento di frammenti di cromatina della dimensione di 100 kb utilizzando come input i contatti della corrispondente matrice. È prevista la possibilità di settare molti parametri, trac cui la distanza di compenetrazione tra due perle, la massima distanza che due perle (i loro baricentri) possono avere (nel rispetto delle dimensioni del nucleo della cellula), l'ampiezza massima delle perturbazioni elementari, il numero di passi da effettuare, la massima curvatura ammessa. L'elenco completo è dato al Paragrafo 7. Dopo aver definito le variabili locali e globali, la procedura calcola i diametri di ogni perla assumendo che esista una relazione inversa tra il numero di contatti che il relativo bin ha con se stesso (valore sulla diagonale della matrice dei contatti) e l'ingombro del relativo frammento di DNA nello spazio (in pratica, più un bin avrà contatti interni, più è lecito pensare che si trovi in una configurazione "aggrovigliata" e meno ingombrante). Successivamente la procedura filtra i picchi della matrice ed individua quelli (non appartenenti alla diagonale) che superano una certa soglia. Questi vengono selezionati e la loro posizione nella matrice viene salvata, vale a dire, vengono localizzate e memorizzate le coppie di perle associate: se p.es. uno dei picchi selezionati è nella posizione $(3, 9)$, significa che le basi appartenenti ai bin 3 e 9 si toccano spesso). Ogni coppia di perle possiede una "energia iniziale", vale a dire una distanza iniziale al passo 0 (per scelta, tale distanza è la massima possibile: la configurazione iniziale vede infatti tutte le perle allineate sull'asse delle ascisse, con angoli planari e diedrali nulli). Perturbando in modo gaussiano gli angoli planari, gli angoli diedrali e le distanze tra i vertici della catena, si ottengono configurazioni associate ognuna a un valore di energia, dato dalla somma tra le distanze euclidee relative alle coppie di loci selezionate filtrando i picchi della matrice. L'obiettivo è far avvicinare le coppie di perle che nella matrice dei contatti risultano toccarsi molte volte, lasciando le altre libere di muoversi nel rispetto dei vincoli assegnati. Le distanze tra i vertici della catena non possono essere perturbate incondizionatamente, ma potranno assumere un valore minimo e uno massimo; se una di queste due soglie viene oltrepassata, la perturbazione cambia segno. Anche per gli angoli planari esiste un vincolo: la curvatura della catena non può superare una certa soglia scelta inizialmente. Gli angoli diedrali, al contrario, sono liberi di ruotare entro l'intero angolo giro.

L'algoritmo scelto per la perturbazione è una versione semplificata di annealing simulato, descritta più in dettaglio al Paragrafo 7 per la particolare implementazione qui utilizzata. Dopo un numero di passi prestabilito, l'algoritmo si ferma, e la configurazione corrente della catena viene assunta come configurazione finale.

La prima procedura produce un numero arbitrario di frammenti di catena con conformazioni che rispettano la frequenza dei contatti della matrice binnata a 100 kb. Nei miei esperimenti, ho scelto pezzi di catena lunghi esattamente 10 bin, in modo da ottenere pezzi già utilizzabili nella seconda procedura, che unisce e perturba pezzi di catena sfruttando i dati della matrice binnata a 1Mb.

Osservazioni Le matrici grezze di Lieberman-Aiden binnate a 100 kb presentano blocchi sulla diagonale principale che identificano settori con contatti interni più frequenti. Tali blocchi hanno una lunghezza genomica variabile, compresa all'incirca tra 10 e 20 elementi, e saranno sfruttati per individuare pezzi di catena da analizzare separatamente, come se fossero compartimenti differenti del filamento. Nella prima procedura non si tiene conto dei contatti tra bin appartenenti a pezzi di catena differenti, ma questo dato viene poi valutato nella seconda procedura, che costruisce l'obiettivo dai dati alla risoluzione di 1Mb.

5.3 Seconda procedura

La seconda procedura prende come dati iniziali i pezzi di catena output della prima procedura. Tali frammenti all'inizio sono slegati tra loro, il loro vertice iniziale è, per tutti i frammenti, situato nell'origine. Per poterli unire e formare una nuova catena alla successiva risoluzione, ho valutato il loro ingombro, considerando il loro centroide e le distanze di esso dal punto iniziale e finale della catena. Per poter creare degli assi di rotazione attorno ai quali frammenti consecutivi possano ruotare (e che passino per il loro punto di contatto), ho considerato il segmento che unisce due centroidi consecutivi, costituito dall'allineamento dei due segmenti che uniscono i centroidi con l'ultimo punto di una catena, che coincide con il punto iniziale della catena successiva. I due raggi interni di un ingombro (quello che unisce punto iniziale e centroide e quello che unisce centroide e il punto finale) possiedono un angolo interno, detto angolo planare, e identificano un piano. Ogni elemento della nuova catena ha così associato un piano. Ottenuti gli angoli diedrali che questi piani formano tra loro, si hanno tutti gli elementi utili per costruire la catena, vale a dire, le distanze tra i centri, gli angoli planari e gli angli diedrali. Nella seconda procedura, le lunghezze dei vettori che uniscono i baricentri delle sfere e i loro angoli planari sono fissi, perché dipendono da proprietà geometriche dei frammenti di catena già calcolati dalla prima procedura. Possono però essere perturbati gli angoli diedrali, che rappresentano la rotazione con cui ogni perla si posiziona rispetto alla precedente. Tale rotazione, a priori, non possiede valori preferenziali, ma usando la funzione obiettivo ottenuta come nella prima procedura dalla matrice dei contatti a 1 Mb è possibile trovare configurazioni ottimali.

5.4 Ricostruzione della catena e visualizzazione dei risultati

Alla fine del ciclo, il codice produce il ripiegamento delle due catene, alle risoluzioni di 100 kb (locale) e 1 Mb (globale). L'ultimo passaggio consiste nel mettere insieme i dati provenienti dai due ripiegamenti, in modo da ottenere le coordinate di tutti i punti delle catene locali all'interno della struttura della catena globale.

6 Osservazioni sul metodo

6.1 Frattalità

il metodo permette di affrontare il problema in un'ottica multiscala, ricostruendo prima frammenti piccoli e poi frammenti più grandi che li contengono. Questo approccio permette di affrontare problemi biologici nell'ottica più verosimile possibile, considerando problemi a livello di intere strutture, senza perdere caratteristiche relative ai dettagli locali.

6.2 Salvaguardia delle caratteristiche fisico-chimiche

L'approccio modellistico permette non solo di mantenere sempre e comunque caratteristiche fisico-chimiche appartenenti al sistema biologico, ma anche di escludere configurazioni non accettabili dal punto di vista fisico (non permettendo collisioni, compenetrazioni, allontanamenti e curvature eccessive). Un altro aspetto importante di questo approccio è quello di lasciare aperta la porta all'inserimento di ulteriore conoscenza a priori.

6.3 Utilizzo dei quaternioni

Il fatto di utilizzare i quaternioni invece delle coordinate cartesiane, oltre allo snellimento computazionale, permette di considerare il sistema preso in esame come indipendente da un sistema di riferimento assoluto. In questo modo, è molto più facile analizzare separatamente i frammenti, perturbandoli e modificando la loro configurazione, per poi andare a ricomporli successivamente a livello globale.

6.4 Prospettive di raffinamento

È possibile pensare di partizionare la matrice in bin di estensioni differenti tra loro. Questa possibilità è particolarmente interessante alla luce dei risultati in [7], che indicano l'esistenza di compartimenti più compatti all'interno della catena della cromatina. Inoltre, è necessario riflettere sul fatto che un cromosoma possiede parti da trattare con maggiore attenzione, come i telomeri ed i centromeri, con aree più o meno ricche di geni e sequenze ripetute.

6.5 Dati di input

I risultati qui riportati utilizzano i dati grezzi forniti in [1], senza nessuna preelaborazione per la ripulitura dai bias. Fermo restando che sarà comunque necessario ripetere le prove utilizzando dati in qualche modo corretti, si deve considerare che l'algoritmo utilizza solo i picchi della matrice dei contatti, indipendentemente dalla loro entità, per cui la ripulitura potrebbe essere un aspetto meno critico di quanto non sia quando si ricorre alla trasformazione contatti–distanze.

7 Risultati

Come anticipato, l'algoritmo di stima consente di settare molti parametri. Eccone l'elenco completo:

- nbinkb: Numero di bin di 100kb in ogni frammento di catena (prima procedura)
- nbinMb: Numero di bin di 1Mb in ogni frammento di catena (seconda procedura)
- numiterationskb: Numero di iterazioni nella prima procedura
- numiterationsMb: Numero di iterazioni nella seconda procedura
- mincontactskb: Numero minimo di contatti per considerare vicini due bin nella prima procedura
- Numcontkb: Numero di contatti utilizzati nel calcolo dell'energia per la prima procedura
- NumcontMb: Numero di contatti utilizzati nel calcolo dell'energia per la seconda procedura
- diagskb: Numero di diagonali centrali trascurate nel calcolo dell'energia per la prima procedura (matrice binnata a 100kb)
- diagsMb: Numero di diagonali centrali trascurate nel calcolo dell'energia per la seconda procedura (matrice binnata a 1Mb)
- mindistkb: Distanza di compenetrazione per i bin della prima procedura (nanometri)
- maxdistkb: Distanza massima ammissibile per i bin della prima procedura (nanometri)
- maxangle: Massimo angolo di curvatura ammissbile per entrambe le catene (gradi)
- Perturbazione della distanza tra i centri dei bin nella prima procedura (nanometri)
- Perturbazione degli angoli planari nella prima procedura (radianti)
- Perturbazione degli angoli diedrali nella prima procedura (radianti)
- Perturbazione degli angoli planari della seconda procedura (radianti)

Le simulazioni sono state svolte con parametri diversi sugli stessi dati, per valutarne l'effetto sulla soluzione. I dati utilizzati sono presi da [1], prime 10Mb del cromosoma 18 (GM06690). I valori dei parametri utilizzati sono elencati in dettaglio nelle tabelle relative ai singoli esperimenti. Ne sono stati utilizzati due insiemi fissati: uno per gli esperimenti 1 e 2, che hanno sfruttato la matrice data, privata della diagonale principale e due subdiagonali per lato, e uno per gli esperimenti 3, 4 e 5, cha hanno invece trattato la matrice privata della diagonale principale e una sola subdiagonale per lato. In Figura 4, sono visualizzate le matrici utilizzate nei due gruppi di esperimenti. I dati originali coincidono con le prime 100 righe e 100 colonne della matrice a sinistra in Figura 1; i dati sulla diagonale sono azzerati, e la prima procedura di ricostruzione si applica ai dieci blocchi diagonali 10×10 evidenziati in figura. Su ognuno di questi blocchi viene selezionato un numero opportuno di massimi nel numero di contatti (cinque per gli esperimenti 1 e 2, e dieci per gli esperimenti 3, 4 e 5), da cui si costruisce la

funzione obiettivo. Questi massimi sono evidenziati dai punti rossi all'interno di ogni blocco diagonale. Terminata le prima procedura, ogni blocco 10×10 diviene un elemento della matrice a risoluzione 1 Mb su cui prosegue la ricostruzione. Anche su questa nuova matrice viene azzerata la diagonale e vengono selezionati un certo numero di picchi, evidenziati in figura con i cerchi di colore arancione. Gli esperimenti 1 e 2 assumono una distanza minima di 100 nm per i bin di 100 kb e permettono una curvatura massima di 120 gradi, mentre gli esperimenti 3, 4 e 5 ammettono una distanza minima di 70 nm e una curvatura massima di 140 gradi. In tutti i casi, la funzione energia è formata come somma delle distanze euclidee tra le coppie di loci selezionate. La procedura Monte Carlo, una sorta di annealing simulato semplificato, genera transizioni casuali accettate in probabilità con la consueta strategia Metropolis, sulla base della funzione energia divisa per un parametro temperatura, con valore iniziale 1000 e legge di raffreddamento $T_{i+1} = 0.998T_i$, dove l'indice i rappresenta ogni singola proposta di transizione. Il fatto che due esperimenti siano condotti con gli stessi parametri non implica il raggiungimento di soluzioni identiche, perché le perturbazioni apportate alla configurazione corrente durante l'iterazione sono casuali (gaussiane) e la funzione obiettivo non risente della configurazione dell'intera catena, ma soltanto di alcuni bin; le rimanenti perle hanno pertanto la possibilità di variare la loro posizione nello spazio nel solo rispetto dei vincoli chimico-fisici. Come si può notare dai risultati degli esperimenti, le prime due simulazioni producono conformazioni meno compatte, mentre le simulazioni 3, 4 e 5 permettono ripiegamenti maggiori, e dunque configurazioni finali più concentrate. Nei successivi sottoparagrafi si riportano i risultati di tutti e cinque gli esperimenti condotti. Per ognuno di loro si riportano anche i grafici dell'andamento dell'energia con il numero di iterazioni per i 10 frammenti con risoluzione a 100 kb e per la ricostruzione globale a 1 Mb di risoluzione. Da questi grafici, associati agli andamenti delle transizioni accettate con aumento e con diminuzione di energia, si vede che nessuno degli esperimenti condotti si concluso con il raggiungimento di una condizione di equilibrio. I risultati qui mostrati devono quindi essere presi come una dimostrazione di fattibilità, piuttosto che come risultati da confrontare con conoscenza biologica disponibile. Detto questo, si nota comunque che le configurazioni ottenute sono sempre coerenti con le nostre richieste. Alcune caratteristiche del ripiegamento, poi, sono comuni a tutte le configurazioni: per esempio, la parte centrale, dove si trova la maggior parte dei contatti, risulta sempre più compatta, mentre la parte iniziale possiede meno contatti (trattandosi di un telomero) e resta quindi un po' più distesa. I diametri degli elementi di catena, calcolati come spiegato al paragrafo 5.2, sono mostrati in Figura 5.

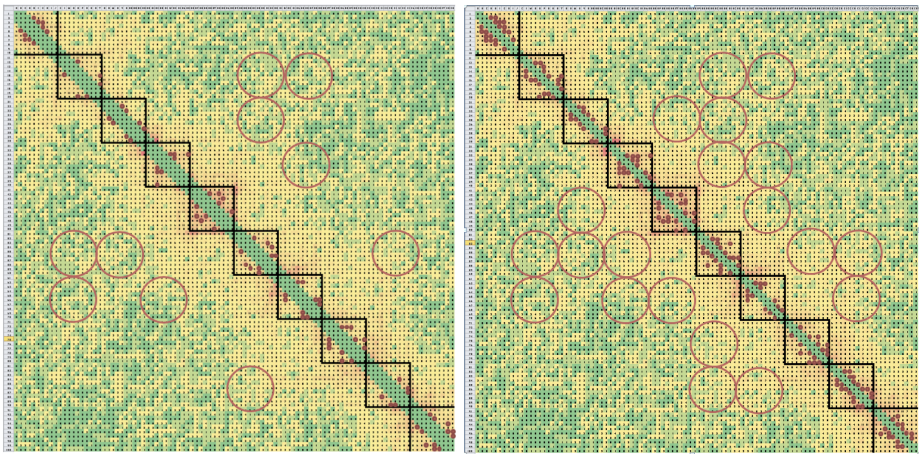

Figura4. Matrici di contatto usate per gli esperimenti 1 e 2 (a sinistra), e 3, 4, e 5 (a destra). In ognuna sono evidenziati i frammenti lunghi 1 Mb (blocchi diagonali 10×10), i picchi con valore al disopra della soglia su ognuno di essi (punti rossi), e i picchi con valore al disopra della soglia tra gli elementi della matrice a risoluzione ridotta (cerchi arancione). Scala di colore come in Figura 1.

	Framm 1	Framm 2	Framm 3	Framm 4	Framm 5	Framm 6	Framm 7	Framm 8	Framm 9	Framm 10
Bin 1	406,1	435,5	424,4	419	388,4	415,1	391,4	432,2	419,3	426,5
Bin 2	432,5	440,6	435,2	395,3	389,9	423,5	413,9	437,6	414,8	434
Bin 3	386,9	445,4	420,5	411,8	388,4	417,5	415,1	435,8	407,6	425,3
Bin 4	396,2	443	420,5	413	423,8	425,9	410	420,5	411,8	414,8
Bin 5	409,4	444,5	425,9	409,7	396,5	417,8	408,8	421,1	419	406,4
Bin 6	413,3	448,4	402,2	415,7	413,9	422,3	435,2	432,5	415,7	412,4
Bin 7	423,8	434,9	424,1	412,4	390,5	398	434,3	408,8	401,6	408,5
Bin 8	425,6	429,2	428,6	379,4	423,8	407,9	417,8	384,2	401,9	409,4
Bin 9	416	412,1	373,1	401	397,4	397,1	417,5	422	385,1	409,1
Bin 10	409,7	428,9	413,6	393,5	399,8	421,7	436,1	416	403,7	380

Figura5. Diametri, in nm, degli elementi di catena che modellano il tratto di cromatina considerato, calcolati in base al numero di contatti di ogni bin con se stesso.

7.1 Esperimento 1

Tabella1. Parametri Esperimento 1

nbinkb	10
nbinMb	10
numiterationskb	2000
numiterationsMb	2000
mincontactskb	15
numcontkb	5
numconMb	5
diagskb	3
diagsMb	3
mindistkb	100
maxdistkb	8000
maxangle	120
Perturbation vector	[-10 ... 10]
Perturbation planar	[-0.1 ... 0.1]
Perturbation dihedral 1	[-0.1 ... 0.1]
Perturbation dihedral 2	[-0.1 ... 0.1]
Execution time (h)	21

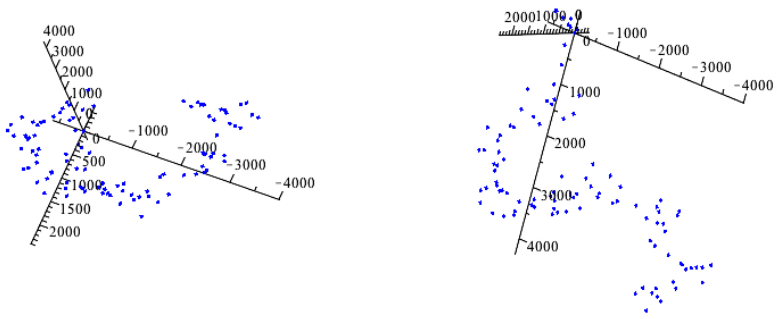

Figura6. Risultati esperimento 1.

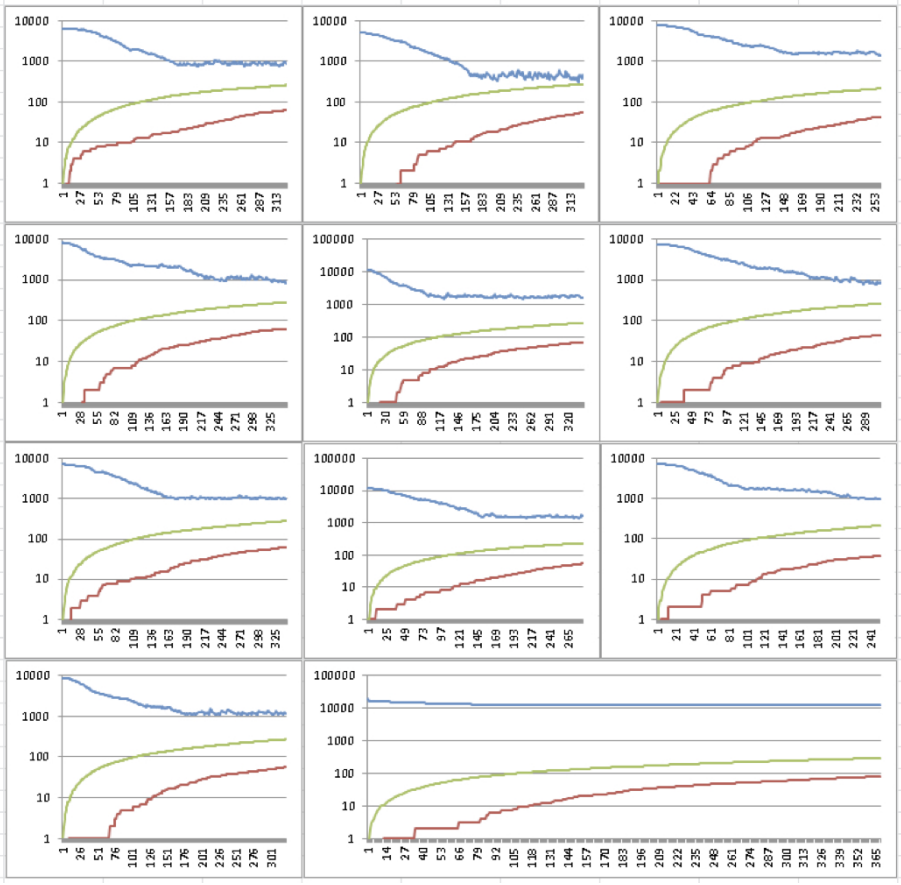

Figura7. Andamenti dell'energia (blu) e contatori del numero di transizioni accettate con diminuzione di energia (verde) e con aumento di energia (rosso) nell'esperimento 1, per i dieci frammenti ricostruiti con la prima procedura e per la ricostruzione dell'intera catena lunga 10 Mb con la seconda procedura (ultimo grafico in basso a destra).

7.2 Esperimento 2

Tabella2. Parametri Esperimento 2

nbinkb	10
nbinMb	10
numiterationskb	2000
numiterationsMb	2000
mincontactskb	15
numcontkb	5
numconMb	5
diagskb	3
diagsMb	3
mindistkb	100
maxdistkb	8000
maxangle	120
Perturbation vector	[-10 ... 10]
Perturbation planar	[-0.1 ... 0.1]
Perturbation dihedral 1	[-0.1 ... 0.1]
Perturbation dihedral 2	[-0.1 ... 0.1]
Execution time (h)	21

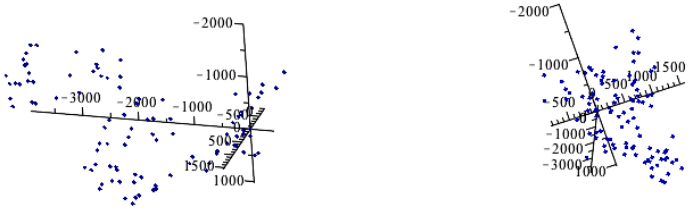

Figura8. Risultati esperimento 2.

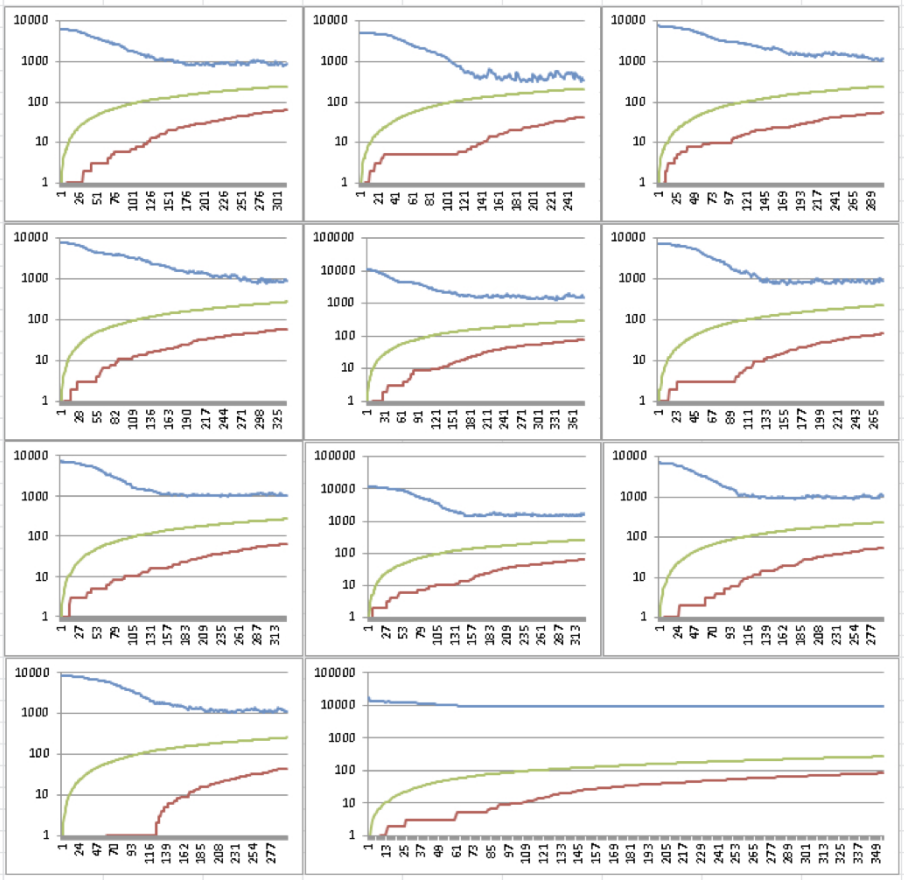

Figura9. Andamenti dell'energia (blu) e contatori del numero di transizioni accettate con diminuzione di energia (verde) e con aumento di energia (rosso) nell'esperimento 2, per i dieci frammenti ricostruiti con la prima procedura e per la ricostruzione dell'intera catena lunga 10 Mb con la seconda procedura (ultimo grafico in basso a destra).

7.3 Esperimento 3

Tabella3. Parametri Esperimento 3

nbinkb	10
nbinMb	10
numiterationskb	2000
numiterationsMb	2000
mincontactskb	15
numcontkb	10
numconMb	10
diagskb	2
diagsMb	2
mindistkb	70
maxdistkb	8000
maxangle	140
Perturbation vector	[-10 ... 10]
Perturbation planar	[-0.1 ... 0.1]
Perturbation dihedral 1	[-0.1 ... 0.1]
Perturbation dihedral 2	[-0.1 ... 0.1]
Execution time (h)	21

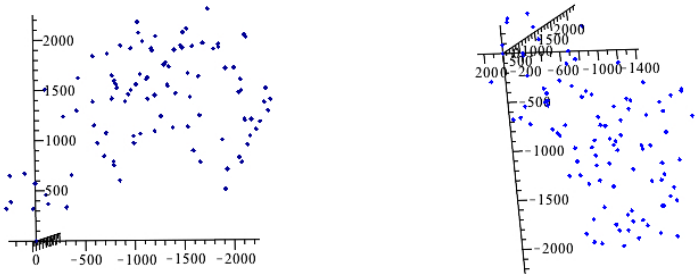

Figura10. Risultati esperimento 3.

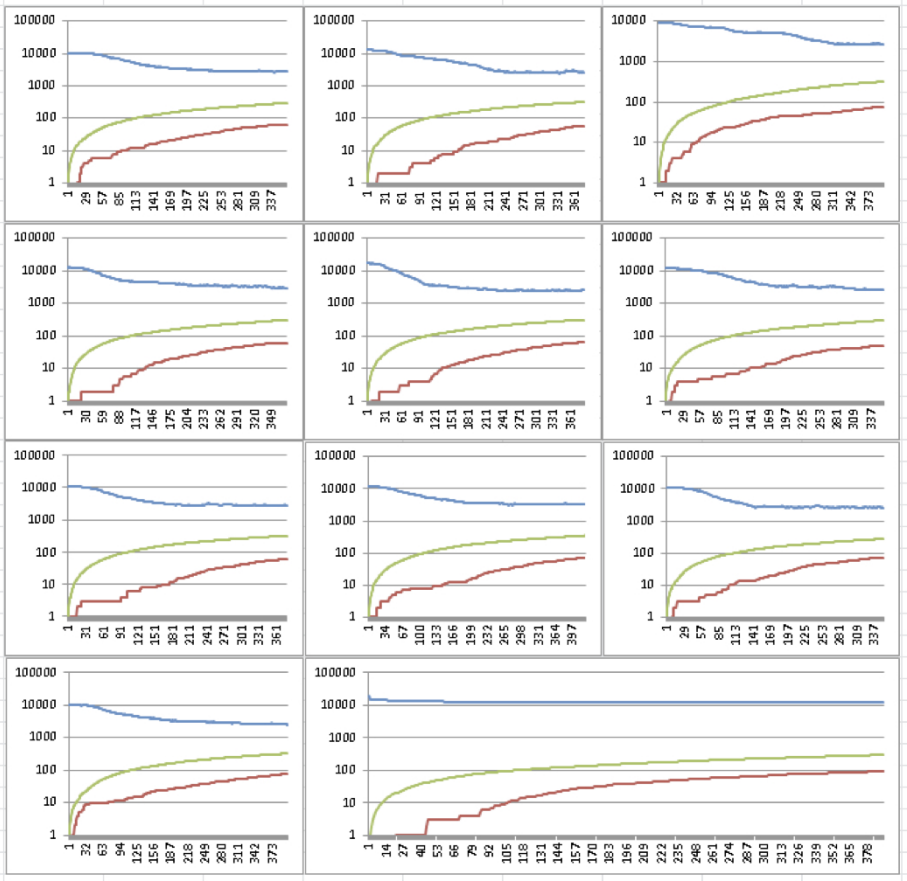

Figura11. Andamenti dell'energia (blu) e contatori del numero di transizioni accettate con diminuzione di energia (verde) e con aumento di energia (rosso) nell'esperimento 3, per i dieci frammenti ricostruiti con la prima procedura e per la ricostruzione dell'intera catena lunga 10 Mb con la seconda procedura (ultimo grafico in basso a destra).

7.4 Esperimento 4

Tabella4. Parametri Esperimento 4

nbinkb	10
nbinMb	10
numiterationskb	2000
numiterationsMb	2000
mincontactskb	15
numcontkb	10
numconMb	10
diagskb	2
diagsMb	2
mindistkb	70
maxdistkb	8000
maxangle	140
Perturbation vector	[-10 ... 10]
Perturbation planar	[-0.1 ... 0.1]
Perturbation dihedral 1	[-0.1 ... 0.1]
Perturbation dihedral 2	[-0.1 ... 0.1]
Execution time (h)	21

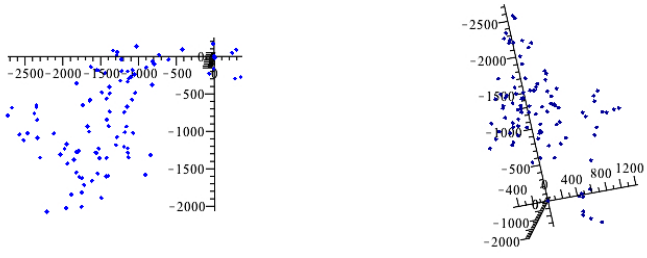

Figura12. Risultati esperimento 4.

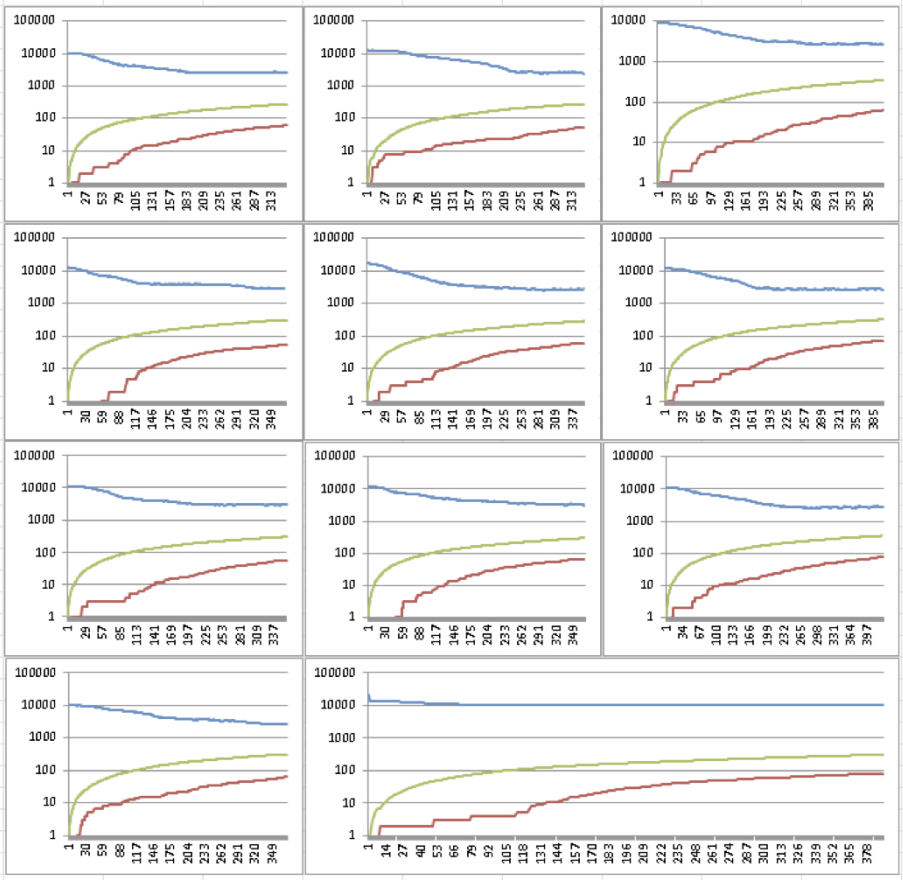

Figura13. Andamenti dell'energia (blu) e contatori del numero di transizioni accettate con diminuzione di energia (verde) e con aumento di energia (rosso) nell'esperimento 4, per i dieci frammenti ricostruiti con la prima procedura e per la ricostruzione dell'intera catena lunga 10 Mb con la seconda procedura (ultimo grafico in basso a destra).

7.5 Esperimento 5

Tabella5. Parametri Esperimento 5

nbinkb	10
nbinMb	10
numiterationskb	2000
numiterationsMb	2000
mincontactskb	15
numcontkb	10
numconMb	10
diagskb	2
diagsMb	2
mindistkb	70
maxdistkb	8000
maxangle	140
Perturbation vector	[-10 ... 10]
Perturbation planar	[-0.1 ... 0.1]
Perturbation dihedral 1	[-0.1 ... 0.1]
Perturbation dihedral 2	[-0.1 ... 0.1]
Execution time (h)	21

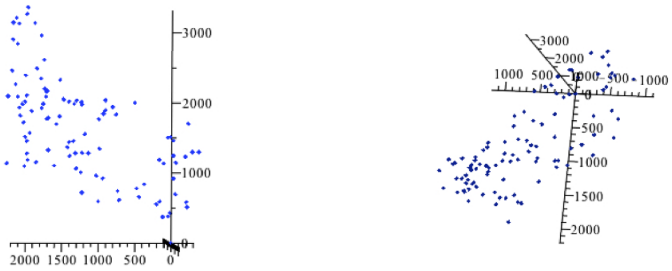

Figura14. Risultati esperimento 5.

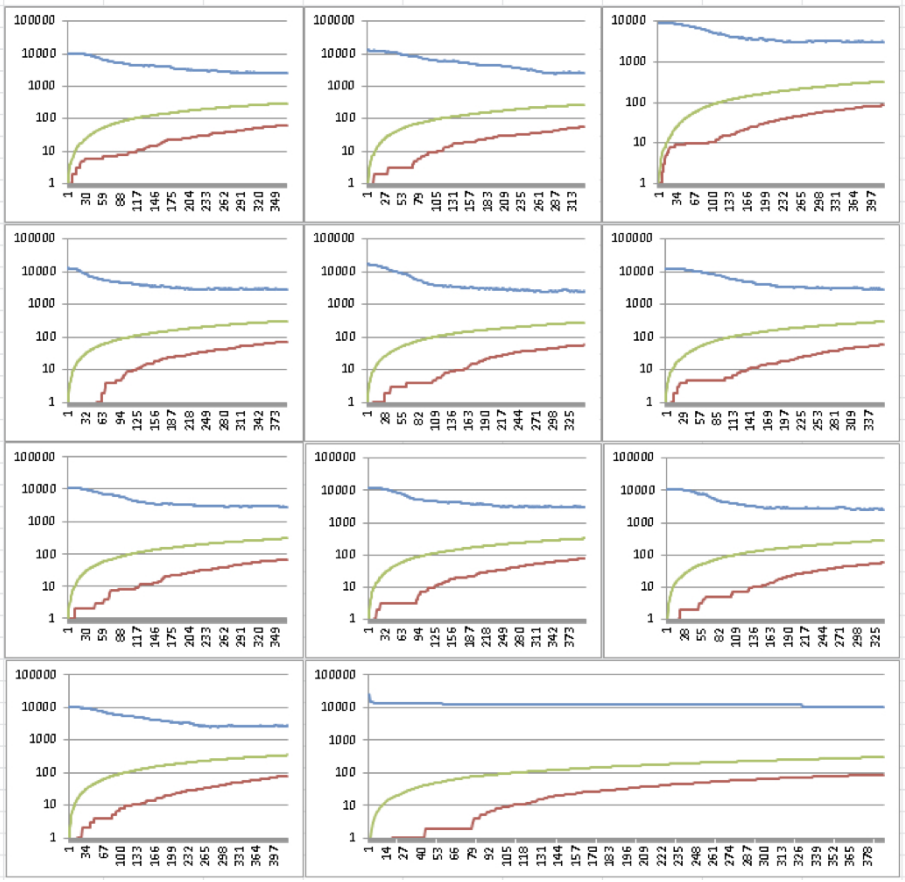

Figura15. Andamenti dell'energia (blu) e contatori del numero di transizioni accettate con diminuzione di energia (verde) e con aumento di energia (rosso) nell'esperimento 5, per i dieci frammenti ricostruiti con la prima procedura e per la ricostruzione dell'intera catena lunga 10 Mb con la seconda procedura (ultimo grafico in basso a destra).

8 Conclusione

In questo rapporto viene proposta una strategia per la ricostruzione della struttura della cromatina da dati di contatto ottenuti da esperimenti Hi-C. Gli aspetti originali della proposta riguardano innanzitutto il trattamento dei dati iniziali. A differenza delle altre strategie proposte, le frequenze di contatto vengono utilizzate senza essere prima tradotte in distanze euclidee, e assumendo che solo le coppie di frammenti molto frequentemente a contatto risultino stabilmente vicine in pressoché tutte le configurazioni assunte dalle cellule esaminate. In secondo luogo, l'approccio qui proposto è intrinsecamente multiscala, in quanto può sfruttare dati a tutte le risoluzioni. Non dovendo passare da contatti a distanze,

l'algoritmo consente infatti di procedere alla stima anche a quelle risoluzioni per cui le matrici dei contatti presentano elementi nulli. Al limite, partendo dalla risoluzione originaria dell'esperimento Hi-C, si possono raggruppare i dati in bin di dimensione diversa, eventualmente anche in base a conoscenze funzionali già disponibili,[2] ricostruire sottocatene della lunghezza voluta, e quindi passare alla scala successiva sfruttando le configurazioni già calcolate e una nuova binnatura della matrice. Questa procedura può essere iterata fino ad arrivare alla massima scala desiderata. Gli esperimenti qui riportati sono solo tesi a mostrare la fattibilità dell'approccio proposto. Dovranno continuare prendendo in considerazione tratti opportuni di DNA, su cui sia possibile condurre una validazione biologica dei risultati. La parte innovativa della proposta è quella che riguarda il modello dati, mentre l'algoritmo di ricerca è un Monte Carlo standard (oltretutto, in versione semplificata). La prosecuzione della ricerca dovrà anche verificare la possibilità di sfruttare le peculiarità del problema in esame per sviluppare algoritmi più efficaci e cercare di ottimizzarli dal punto di vista della complessità computazionale e l'implementabilità su architetture ad alte prestazioni.

Riferimenti bibliografici

1. Lieberman-Aiden, E, et al. (2009): Comprehensive mapping of long-range interactions reveals folding principles of the human genome. Science 326: 289–293.
2. Yaffe, E, Tanay, A (2011): Probabilistic modeling of Hi-C contact maps eliminates systematic biases to characterize global chromosomal architecture. Nat Genet 43: 1059–1065.
3. Hu, M, et al. (2013): Bayesian inference of spatial organizations of chromosomes. PLoS Comput. Biol., 9, e1002893.
4. Rousseau, M, et al. (2011): Threedimensional modeling of chromatin structure from interaction frequency data using Markov chain Monte Carlo sampling. BMC Bioinformatics 12: 414.
5. Hu, M, et al. (2013b): Understanding spatial organization of chromosomes via statistical analysis of Hi-C data. Quantitative Biology 1: 156–174.
6. Meluzzi, D, Arya, G (2013): Recovering ensembles of chromatin conformations from contact probabilities. Nucleic Acid Research, 41, 63-75.
7. Dixon J, et al. (2012): Topological domains in mammalian genomes identified by analysis of chromatin interactions. Nature 485: 376–380.

[2] per esempio, elementi genici promotori, enhancers, ecc.

A Appendice: Codice Maple

```
> restart;
> with(LinearAlgebra);
> with(Typesetting);
> with(ArrayTools);
> with(ListTools);
> with(plots);
> with(geom3d);
> with(Statistics);
> Settings(functionassign = false);
> interface(displayprecision = 4);
> with(Quaternions);
> m := 200;
> nbinkb := 10;
> nbinMb := 10;
> numiterationskb := 10;
> numiterationsMb := 10;
> mincontactskb := 15;
> numcontkb := 5;
> numcontMb := 5;
> diagskb := 3;
> diagsMb := 3;
> mindistkb := 100;
> maxdistkb := 8000;
> maxangle := 120;
> alpha0 := readdata("C:/users/claudia/Desktop/Maple-  Chromatin/Chain-Maple/
ini_values/alpha.txt", m-2);
> alpha0 := convert(alpha0, Vector);
> beta0 := readdata("C:/users/claudia/Desktop/Maple-Chromatin/Chain-Maple/
ini_values/beta.txt", m-2);
> beta0 := convert(beta0, Vector);
> HM := readdata("C:/users/claudia/Desktop/Data Hi-C/Data L-   A/
binned100Kb_obs_txt/chr18.txt", m);
> HM := convert(HM, Matrix);
> arad := evalf(convert(maxangle*degrees, radians));
> rot := proc (v, q) options operator, arrow; `<,>`((2*q[1]^2-1+2*q[2]^2)*v[1]+
(2*q[2]*q[3]-2*q[1]*q[4])*v[2]+(2*q[2]*q[4]+2*q[1]*q[3])*v[3], (2*q[2]*q[3]+
2*q[1]*q[4])*v[1]+(2*q[1]^2-1+2*q[3]^2)*v[2]+(2*q[3]*q[4]-2*q[1]*q[2])*v[3],
(2*q[2]*q[4]-2*q[1]*q[3])*v[1]+(2*q[3]*q[4]+2*q[1]*q[2])*v[2]+(2*q[1]^2-1+
2*q[4]^2)*v[3]) end proc;
> fincoords := []; findists := []; collisionskb := []; stepsenkb := [];
> a := rand(-10 .. 10);
segm:=proc(numchrom::integer,frag::integer,n1::integer,n2::integer,
mincontacts::integer,numcontkb::integer,diags::integer,mindist::integer,
maxdist::integer,numiterations::integer)
local MD,diam,vect,pHM, pHMM,inienergy;
local l,t,s, no,vectinizkb, HM2, n,minkb;
local M,c,cc,energy,vt,v1,al,be,Mcoord,temp;
local coords, b, g, en, den, pen;
global fincoord, collisionkb,stepenkb;
n:=n2-n1+1:
MD:=Matrix():
diam:=Vector(n):
vect:=Vector(n-1):
pHM:=[]:
pHMM:=[]:
collisionkb:=[]:
stepenkb:=[]:
fincoord:=Matrix(n,3):
 for l from 1 to n do
   diam(l):=500-(HM(n1+l,n1+l)*300)/(1000);
 end do:
 for l from 1 to (n-1) do
   vectinizkb(l):=(diam(l))/(2)+(diam(l+1))/(2);
   vect(l):=(diam(l))/(2)+(diam(l+1))/(2);
 end do:
 for l from 1 to (n-2) do
```

```
  alpha(l):=alpha0(l):
  beta(l):=beta0(l):
end do:
for l from 1 to n do
  for t from l to n do
    if t=l then
      MD(t,l):=0:
    else
      MD(t,l):= add(vect(s),s=l..t-1):
      MD(l,t):=MD(t,l):
    end if:
  end do:
end do:
for l from 1 to n-1 do
  for s from 1 to n-1 do
    if (s-l)>=diags then
      pHM:=[HM(n1+l,n1+s),op(pHM)];
    end if:
  end do:
end do:
pHM:=Sort(pHM,order=descending):
 if pHM[1]<=mincontacts then
   minkb:=pHM[1]:
 else
 if nops(pHM)>=numcontkb then
   minkb:=pHM[numcontkb]:
 else
   minkb:=pHM[nops(pHM)]:
 end if:
end if:
for l from 1 to n-1 do
  for s from 1 to n-1 do
    if (HM(n1+l,n1+s)>=minkb and (s-l)>=diags) then
      lprint('nel frammento',frag,'del cromosoma',numchrom,
      'ci  sono',HM(n1+l,n1+s),'contatti fra il bin', n1+l,'e il bin',n1+s):
      pHMM:=[<HM(n1+l,n1+s),l+1,s+1>,op(pHMM)];
    end if:
  end do:
end do:
no:=nops(pHMM);
inienergy:=0:
for l from 1 to no do
  inienergy:=inienergy+MD(pHMM[l][2],pHMM[l][3]):
end do:
lprint('energia iniziale',inienergy):
stepenkb:=[<frag,inienergy,0,1>]:
M:=Matrix(n,n):
c:=Vector(numiterations):
cc:=Vector(numiterations):
energy:=Vector(numiterations):
vt:=Vector(n-1):
v1:=Vector(n-1):
al:=Vector(n-2):
be:=Vector(n-2):
Mcoord:=Matrix(n,3):
temp:=1000:
for s from 1 to numiterations do
  coords[s]:=convert(cat("Chrom",numchrom,"Binkb",frag,"Coords",s),name):
 temp:=temp*0.998:
 for t from 1 to n-1 do
   b:=a():
   if vectinizkb(t)-20<=vect(t)+evalf(b)<=vectinizkb(t)+20 then
     vt(t):=vect(t)+evalf(b)
   else
     vt(t):=vect(t)-evalf(b)
   end if:
 end do:
 for t from 1 to n-2 do
```

```
    b:=(a())/(100):
    if -arad<= evalf(alpha(t))+evalf(b)<=arad then
      al(t):=alpha(t)+evalf(b)
    else
      al(t):=alpha(t)-evalf(b)
    end if:
  end do:
  for t from 1 to n-2 do
    b:=(a())/(100):
    be(t):=beta(t)+evalf(b):
  end do:
#‘CALCOLO DELLE ROTAZIONI DOVUTE AGLI ANGOLI PLANARI‘
  p[al](1):=<0,0,0>:p[al](2):=<vt(1),0,0>:
  alp(1):=al(1):
  for t from 3 to n do
    alp(t-2):=evalf(add(al(x),x=1..t-2)):
    p[al](t):=<p[al](t-1)[1]+evalf((cos(alp(t-2)))*vt(t-1)),p[al](t-1)[2]+
    evalf((sin(alp(t-2)))*vt(t-1)),0>: d(t):=<p[al](t)(1)-p[al](t-1)(1),p[al](t)(2)-
    p[al](t-1)(2),p[al](t)(3)-p[al](t-1)(3)>: normpa(t,s):=norm(d(t),2);
  od:
  #‘CALCOLO DELLE ROTAZIONI DOVUTE AGLI ANGOLI DIEDRALI (CON I  QUATERNIONI)‘
  pr(1):=p[al](1):pr(2):=p[al](2):
  qua(1):=1:qua(2):=1:
  for t from 2 to n do
    v1(t-1):=<p[al](t)[1]-p[al](t-1)[1],p[al](t)[2]-p[al](t-1)[2],p[al](t)[3]-
    p[al](t-1)[3]>:
    v(t-1):=<(p[al](t)[1]-p[al](t-1)[1])/(vt(t-1)),
    (p[al](t)[2]-p[al](t-1)[2])/(vt(t-1)),
    (p[al](t)[3]-p[al](t-1)[3])/(vt(t-1))>:
  od:
  for t from 3 to n do
    q(t):=Qunit(cos(be(t-2)/2)+((v(t-2)[1])*i+(v(t-2)[2])*j+(v(t-2)[3])*k)*
    sin(be(t-2)/2)):
    qua(t):=evalf(qua(t-1)*q(t)):
    p(t):=rot(v1(t-1),<Qscalar(qua(t)),Qicoeff(qua(t)),Qjcoeff(qua(t)),
    Qkcoeff(qua(t))>):
    pr(t):=<p(t)[1]+pr(t-1)[1],p(t)[2]+pr(t-1)[2],p(t)[3]+pr(t-1)[3]>:
  od:
  for t from 2 to n do
    d(t):=<pr(t)[1]-pr(t-1)[1],pr(t)[2]-pr(t-1)[2],pr(t)[3]-pr(t-1)[3]>:
    normpr(t,s):=norm(d(t),2):
  od:
  #‘CALCOLO DELLA MATRICE DELLE DISTANZE‘
  for g from 1 to n do
    for t from 1 to n do
      d(g,t):=<pr(g)[1]-pr(t)[1],pr(g)[2]-pr(t)[2],pr(g)[3]-pr(t)[3]>:
      M(g,t):=norm(d(g,t),2):
    od:
  od:
  for l from 1 to n do
    Mcoord(l,1):=pr(l)(1):
    Mcoord(l,2):=pr(l)(2):
    Mcoord(l,3):=pr(l)(3):
  od:
  c(s):=0:cc(s):=0:
  for g from 1 to n do
    for t from g+1 to n do
      if M(g,t)<=mindist then
        c(s):=c(s)+1:
        lprint(M(g,t),'minimdist',s):
        collisionkb:=[op(collisionkb),<frag,g,t,s>]:
      elif M(g,t)> maxdist then
        cc(s):=cc(s)+1:
        lprint(M(g,t),'maximdist',s):
      end if:
    od:
  od:
  en:=0:
```

```
  for l from 1 to no do
    en:=en+M(pHMM[l][2],pHMM[l][3]):
  end do:
  energy(s):=en:
  #lprint('energia al passo', s,energy(s));
  if s>1 and (c(s)=0 and cc(s)=0)  then
    den:=energy(s)-inienergy:
    #lprint('crom',numchrom,'framm',frag,'ener al passo', s, energy(s),
    'ener iniziale', inienergy,'differ',den):
    if den<=0 then
      inienergy:=energy(s):
      for t from 1 to n-1 do
        vect(t):=evalf(vt(t)):
      end do:
      for t from 1 to n-2 do
        alpha(t):=evalf(al(t)):
        beta(t):=evalf(be(t)):
      end do:
      ExportMatrix(coords[s],Mcoord,target=delimited):
      stepenkb:=[op(stepenkb),<frag,energy(s),den,s>]:
      for l from 1 to n do
        fincoord(l,1):=Mcoord(l,1):
        fincoord(l,2):=Mcoord(l,2):
        fincoord(l,3):=Mcoord(l,3):
      od:
    else
      pen:=exp(-den/(temp)):
      if s=1 or pen>=|10/((a()+0,0001))| then
        inienergy:=energy(s):
        for t from 1 to n-1 do
          vect(t):=evalf(vt(t)):
        end do:
        for t from 1 to n-2 do
          alpha(t):=evalf(al(t)):
          beta(t):=evalf(be(t)):
        end do:
        ExportMatrix(coords[s],Mcoord,target=delimited):
        stepenkb:=[op(stepenkb),<frag,energy(s),den,s>]:
        for l from 1 to n do
          fincoord(l,1):=Mcoord(l,1):
          fincoord(l,2):=Mcoord(l,2):
          fincoord(l,3):=Mcoord(l,3):
        od:
      end if:
     end if:
   end if:
 end do:
end proc:
> #in proc segm va inserito: numchrom, frag, n1, n2, mincontacts,
numcontacts, diags, mindist ,maxdist ,numiterations
> for l from 0 to nbinMb-1 do
    segm(18, l+1, l*nbinkb+1, (l+1)*nbinkb+1, mincontactskb,  numcontkb,
    diagskb, mindistkb, maxdistkb, numiterationskb);
    fincoords := [op(fincoords), fincoord];
    findists := [op(findists), findist];
    collisionskb := [op(collisionskb), collisionkb];
    stepsenkb := [op(stepsenkb), stepenkb]
  end do;
> stepsenkb := convert(stepsenkb, Matrix);
  ExportMatrix("stepsenkb.txt", stepsenkb, target = delimited);
> collisionskb := convert(collisionskb, Matrix);
  ExportMatrix("collisionskb.txt", collisionskb, target = delimited);
> plan := Vector(nbinMb); CHI := Vector(nbinMb);
> B := []; vect1 := []; vect2 := [];
> for l from 1 to nbinMb do
  #‘ BARYCENTERS ‘
  ps:=[]:
  nl:=nops(convert(Column(fincoords[l],1),list));
```

```
  for t from 1 to nl do
    ps:=[op(ps),point(convert(cat("point",t),name),fincoords[l](t,1),
    fincoords[l](t,2),fincoords[l](t,3))]:
  end do:
  B:=[op(B),coordinates(centroid(C,ps))]:
  #VECTORS
  v1:=<B[l,1],B[l,2],B[l,3]>:
  v2:=<fincoords[l](nl,1)-B[l,1],fincoords[l](nl,2)- B[l,2],
  fincoords[l](nl,3)-B[l,3]>:
  vect1:=[op(vect1),norm(v1,2)]:#print(vect1[l],l):
  vect2:=[op(vect2),norm(v2,2)]:#print(vect2[l],l):
  #‘PLANAR ANGLES‘
  vl1:=convert(v1,list):
  vl2:=convert(v2,list):
  plan(l):=sign(a())*Re(evalf(arccos(DotProduct(vl1,vl2)/(vect1[l]*vect2[l])))):
  print(plan(l),planar angle,l):
  #‘DHIEDRAL ANGLES‘
  if l=1 then
    CHI(l):=0:print(CHI(l),dhiedral angle,l):
  else
    vs1:=<B[l-1][1]-fincoords[l-1](1,1),B[l-1][2]-fincoords[l-1](1,2),
    B[l-1][3]-fincoords[l-1](1,3)>:
    vs2:=<fincoords[l](nl,1)-B[l-1][1],fincoords[l](nl,2)-B[l-1][2],
    fincoords[l](nl,3)-B[l-1][3]>:
    vvett1:=<CrossProduct(vs1,vs2)[1],CrossProduct(vs1,vs2)[2],
    CrossProduct(vs1,vs2)[3]>:
    lvett1:=convert(vvett1,list):
    vvett2:=<CrossProduct(v1,v2)[1],CrossProduct(v1,v2)[2],
    CrossProduct(v1,v2)[3]>:
    lvett2:=convert(vvett2,list):
    vsintor:=<CrossProduct(vvett2,vs1)[1],CrossProduct(vvett2,vs1)[2],
    CrossProduct(vvett2,vs1)[3]>:
    lsintor:=convert(vsintor,list):
    sinsin:=DotProduct(lsintor,lvett1):
    if sinsin >= 0   then
      CHI(l):=Re(evalf(arccos(DotProduct(lvett1,lvett2)/
      (norm(vvett1,2)*(norm(vvett2,2))))))
    else
      CHI(l):=Re(-evalf(arccos(DotProduct(lvett1,lvett2)/
      (norm(vvett1,2)*(norm(vvett2,2))))))
    end if:
    print(CHI(l),dhiedral angle,l):
  end if:
 end do:
> HMM := readdata("C:/users/claudia/Desktop/
Data Hi-C/Data L-A/binned1Mb_obs_txt/chr18.txt", m);
> HMM := convert(HMM, Matrix);
> mainchain := proc (numchrom::integer, frag::integer, n1::integer,
n2::integer, numcontMb::integer, diags::integer, numiterations::integer)
  local MD, vect, pHM, pHMM, inienergy;
  local l, t, s, no, vectinizMb, n, minMb;
  local M, c, cc, d, energy, vt, v1, al, be, Mcoord, temp;
  local coords, b, g, en, den, pen;
  global finchi, coordMb, stepenMb;
  n := n2-n1+2;
  MD := Matrix(n);
  vect := Vector(n);
  pHM := [];
  pHMM := [];
  stepenMb := [];
  finchi := Vector(n-1);
  coordMb := Matrix(n+1, 3);
  for l to n do if l = 1 then
    vect(l) := vect1[l];
    vectinizMb(l) := vect1[l]
  elif l = n then
    vect(l) := vect2[l-1];
    vectinizMb(l) := vect2[l-1]
```

```
 else
   vect(l) := vect2[l-1]+vect1[l];
   vectinizMb(l) := vect2[l-1]+vect1[l]
 end if
end do;
for l to n do
  for t from l to n do
    if t = l then
      MD(t, l) := 0
    else
      MD(t, l) := add(vect(s), s = l .. t-1);
      MD(l, t) := MD(t, l)
    end if
  end do
end do;
for l to n-2 do
  for s to n-2 do
    if diags <= s-l then
      pHM := [HMM(n1+l, n1+s), op(pHM)]
    end if
  end do
end do;
pHM := Sort(pHM, order = descending);
if numcontMb <= nops(pHM) then
  minMb := pHM[numcontMb]
else
  minMb := pHM[nops(pHM)]
end if;
for l to n-2 do
  for s to n-2 do
    if minMb <= HMM(n1+l, n1+s) and diags <= s-l then
      lprint('nel*frammento', frag, 'del*cromosoma', numchrom,
      'ci*sono', HMM(n1+l, n1+s), 'contatti*fra*il*bin', n1+l, 'e*il*bin', n1+s);
      pHMM := [<HMM(n1+l, n1+s), l+1, s+1), op(pHMM)>]
    end if
  end do
end do;
no := nops(pHMM);
inienergy := 0;
for l to no do
  inienergy := inienergy+MD(pHMM[l][2], pHMM[l][3])
end do;
lprint('energia*iniziale', inienergy);
stepenMb := [<inienergy, 0, 1>];
M := Matrix(n+1, n+1);
v1 := Vector(n);
c := Vector(numiterations);
cc := Vector(numiterations);
energy := Vector(numiterations);
vt := Vector(n);
al := Vector(n-1);
be := Vector(n-2);
Mcoord := Matrix(n+1, 3);
temp := 1000;
d := Matrix(n+1, n+1);
for t to n do
  vt(t) := vect(t)
end do;
for t to n-1 do
  al(t) := plan(t)
end do;
for t to n-1 do
  beta(t) := CHI(t)
end do;
for s to numiterations do
  coords[s] := convert(cat("Chrom", numchrom, "BinMb", frag,
  "Coords", s), name);
  temp := temp*.998;
```

```
be(1) := beta(1);
for t from 2 to n-1 do
  b := (1/100)*a();
  be(t) := beta(t)+evalf(b)
end do;
p[al](1) := <0, 0, 0>;
p[al](2) := <vt(1), 0, 0>;
for t from 3 to n+1 do
  alp(t-2) := evalf(add(al(x), x = 1 .. t-2));
  p[al](t) := <p[al](t-1)[1]+evalf(cos(alp(t-2))*vt(t-1)),
  p[al](t-1)[2]+evalf(sin(alp(t-2))*vt(t-1)), 0>;
d(t) := <p[al](t)[1]-p[al](t-1)[1], p[al](t)[2]-p[al](t-1)[2],
p[al](t)[3]-p[al](t-1)[3]); normpa(t, s) := norm(d(t), 2>
end do;
pr(1) := p[al](1);
pr(2) := p[al](2);
qua(1) := 1;
qua(2) := 1;
for t from 2 to n+1 do
v1(t-1) := <p[al](t)[1]-p[al](t-1)[1], p[al](t)[2]-p[al](t-1)[2],
p[al](t)[3]-p[al](t-1)[3]>;
v(t-1) := <(p[al](t)[1]-p[al](t-1)[1])/vt(t-1), (p[al](t)[2]-
p[al](t-1)[2])/vt(t-1), (p[al](t)[3]-p[al](t-1)[3])/vt(t-1)>
end do;
for t from 3 to n+1 do
  q(t) := Qunit(cos((1/2)*be(t-2))+(v(t-2)[1]*i+v(t-2)[2]*j+
  v(t-2)[3]*k)*sin((1/2)*be(t-2)));
  qua(t) := evalf(qua(t-1)*q(t));
  p(t) := rot(v1(t-1), <Qscalar(qua(t)), Qicoeff(qua(t)),
  Qjcoeff(qua(t)), Qkcoeff(qua(t))>;
  pr(t) := <p(t)[1]+pr(t-1)[1], p(t)[2]+pr(t-1)[2], p(t)[3]+pr(t-1)[3]>
end do;
for t from 2 to n+1 do
  d(t) := <pr(t)[1]-pr(t-1)[1], pr(t)[2]-pr(t-1)[2], pr(t)[3]-pr(t-1)[3]>;
  normpr(t, s) := norm(d(t), 2)
end do;
for g to n+1 do
  for t to n+1 do
    d(g, t) := <pr(g)[1]-pr(t)[1], pr(g)[2]-pr(t)[2], pr(g)[3]-pr(t)[3]>;
    M(g, t) := norm(d(g, t), 2)
  end do
end do;
for l to n+1 do
  Mcoord(l, 1) := (pr(l))(1);
  Mcoord(l, 2) := (pr(l))(2);
  Mcoord(l, 3) := (pr(l))(3)
end do;
en := 0;
for l to no do
  en := en+M(pHMM[l][2]+1, pHMM[l][3]+1)
end do;
energy(s) := en;
if 1 < s then
  den := energy(s)-inienergy;
  if den <= 0 then
    inienergy := energy(s);
    for t to n do
      vect(t) := evalf(vt(t))
    end do;
    for t to n-1 do
      alpha(t) := evalf(al(t))
    end do;
    for t to n-1 do
      beta(t) := evalf(be(t))
    end do;
    ExportMatrix(coords[s], Mcoord, target = delimited);
    stepenMb := [op(stepenMb), <energy(s), den, s>];
    for l to n+1 do
```

```
        coordMb(l, 1) := Mcoord(l, 1);
        coordMb(l, 2) := Mcoord(l, 2);
        coordMb(l, 3) := Mcoord(l, 3)
      end do
    else
      pen := exp(-den/temp);
      if s = 1 or abs(10/(a(), 1)) <= pen then
        inienergy := energy(s);
        for t to n do
          vect(t) := evalf(vt(t))
        end do;
        for t to n-1 do
          alpha(t) := evalf(al(t))
        end do;
        for t to n-1 do
          beta(t) := evalf(be(t))
        end do;
        ExportMatrix(coords[s], Mcoord, target = delimited);
        stepenMb := [op(stepenMb), <energy(s), den, s)>];
        for l to n+1 do
          coordMb(l, 1) := Mcoord(l, 1);
          coordMb(l, 2) := Mcoord(l, 2);
          coordMb(l, 3) := Mcoord(l, 3)
        end do
      end if
    end if
  end if
end do;
for t to n-1 do
  finchi(t) := beta(t)
end do
end proc;
> #in proc mainchain va inserito: numchrom, frag, n1, n2,
numncontacts, diags, numiterations;
> mainchain(18, 1, 1, nbinMb, numcontMb, diagsMb, numiterationsMb);
> stepenMb := convert(stepenMb, Matrix);
  ExportMatrix("stepenMb.txt", stepenMb, target = delimited);
> for s from 1 to (nbinMb+2) do
    ve(s):=<coordMb(s,1),coordMb(s,2),coordMb(s,3)>:
    #print(ve(s)):
  end do:
> pointplot3d([ve(1), ve(2), ve(3), ve(4), ve(5), ve(6), ve(7), ve(8),
ve(9), ve(10), ve(11), ve(12)],
color = [blue], scaling = constrained, axes = normal);
> #in proc mainchain va inserito: numchrom, frag, n1, n2,
numncontacts, diags, numiterations;
> mainchain(18, 1, 1, nbinMb, numcontMb, diagsMb,numiterationsMb);
> stepenMb := convert(stepenMb, Matrix);
  ExportMatrix("stepenMb.txt", stepenMb, target = delimited);
> for s from 1 to (nbinMb+2) do
    ve(s):=<coordMb(s,1),coordMb(s,2),coordMb(s,3)>:
    #print(ve(s)):
  end do:
> pointplot3d([ve(1), ve(2), ve(3), ve(4), ve(5), ve(6), ve(7), ve(8),
ve(9), ve(10), ve(11), ve(12)],
color = [blue], scaling = constrained, axes = normal);

> pfincoords := []; nb := nbinkb+1;
> for s to nbinMb do
    pfin := Matrix(nb, 3);
    if s = 1 then
      for t to nb do
        v1 := <B[1, 1], B[1, 2], B[1, 3]>;
        v2 := <fincoords[1](nb, 1)-B[1, 1], fincoords[1](nb, 2)-B[1, 2],
        fincoords[1](nb, 3)-B[1, 3]>;
        v3 := CrossProduct(v1, v2); w1 := <ve(2)[1], ve(2)[2], ve(2)[3]>;
        w2 := <(ve(3)[1]-ve(2)[1])*vect2[1]/(vect2[1]+vect1[2]),
        (ve(3)[2]-ve(2)[2])*vect2[1]/
```

```
  (vect2[1]+vect1[2]), (ve(3)[3]-ve(2)[3])*vect2[1]/(vect2[1]+vect1[2])>;
  w3 := CrossProduct(w1, w2);
  M1 := Matrix([v1, v2, v3]);
  M2 := Matrix([w1, w2, w3]);
  MI1 := MatrixInverse(M1);
  L := MatrixMatrixMultiply(M2, MI1);
  pfin(t, 1) := MatrixVectorMultiply(L, <fincoords[1](t, 1),
  fincoords[1](t, 2), fincoords[1](t, 3))>[1];
  pfin(t, 2) := MatrixVectorMultiply(L, <fincoords[1](t, 1),
  fincoords[1](t, 2), fincoords[1](t, 3))>[2];
  pfin(t, 3) := MatrixVectorMultiply(L, <fincoords[1](t, 1),
  fincoords[1](t, 2), fincoords[1](t, 3))>[3]
end do
elif s < nbinMb then
  for t to nb do
    v1 := <B[s, 1], B[s, 2], B[s, 3]>;
    v2 := <fincoords[s](nb, 1)-B[s, 1], fincoords[s](nb, 2)-B[s, 2],
    fincoords[s](nb, 3)-B[s, 3]>;
    v3 := CrossProduct(v1, v2);
    w1 := <(ve(s+1)[1]-ve(s)[1])*vect1[s]/(vect2[s-1]+vect1[s]),
    (ve(s+1)[2]-ve(s)[2])*vect1[s]/
    (vect2[s-1]+vect1[s]), (ve(s+1)[3]-ve(s)[3])*vect1[s]/
    (vect2[s-1]+vect1[s])>;
    w2 := <(ve(s+2)[1]-ve(s+1)[1])*vect2[s]/(vect2[s]+vect1[s+1]),
    (ve(s+2)[2]-ve(s+1)[2])*vect2[s]/
    (vect2[s]+vect1[s+1]), (ve(s+2)[3]-ve(s+1)[3])*vect2[s]/
    (vect2[s]+vect1[s+1])>;
    w3 := CrossProduct(w1, w2);
    M1 := Matrix([v1, v2, v3]);
    M2 := Matrix([w1, w2, w3]);
    MI1 := MatrixInverse(M1);
    L := MatrixMatrixMultiply(M2, MI1);
    pfin(t, 1) := MatrixVectorMultiply(L, <fincoords[s](t, 1),
    fincoords[s](t, 2), fincoords[s](t, 3))>[1]+
    pfincoords[s-1](nb, 1)>;
    pfin(t, 2) := MatrixVectorMultiply(L, <fincoords[s](t, 1),
    fincoords[s](t, 2), fincoords[s](t, 3))>[2]+
    pfincoords[s-1](nb, 2)>;
    pfin(t, 3) := MatrixVectorMultiply(L, <fincoords[s](t, 1),
    fincoords[s](t, 2), fincoords[s](t, 3))>[3]+
    pfincoords[s-1](nb, 3)>
 end do
  else
    for t to nb do
      v1 := <B[s, 1], B[s, 2], B[s, 3]>;
      v2 := <fincoords[s](nb, 1)-B[s, 1], fincoords[s](nb, 2)-B[s, 2],
      fincoords[s](nb, 3)-B[s, 3]>;
      v3 := CrossProduct(v1, v2);
      w1 := <(ve(s+1)[1]-ve(s)[1])*vect1[s]/(vect2[s-1]+vect1[s]),
      (ve(s+1)[2]-ve(s)[2])*vect1[s]/(vect2[s-1]+
      vect1[s]), (ve(s+1)[3]-ve(s)[3])*vect1[s]/(vect2[s-1]+vect1[s])>;
      w2 := <ve(s+2)[1]-ve(s+1)[1], ve(s+2)[2]-ve(s+1)[2],
      ve(s+2)[3]-ve(s+1)[3]>;
      w3 := CrossProduct(w1, w2);
      M1 := Matrix([v1, v2, v3]);
      M2 := Matrix([w1, w2, w3]);
      MI1 := MatrixInverse(M1);
      L := MatrixMatrixMultiply(M2, MI1);
      pfin(t, 1) := MatrixVectorMultiply(L, <fincoords[s](t, 1),
      fincoords[s](t, 2), fincoords[s](t, 3))>[1]+
      pfincoords[s-1](nb, 1)>;
      pfin(t, 2) := MatrixVectorMultiply(L, <fincoords[s](t, 1),
      fincoords[s](t, 2), fincoords[s](t, 3))>[2]+
      pfincoords[s-1](nb, 2)>;
      pfin(t, 3) := MatrixVectorMultiply(L, <fincoords[s](t, 1),
      fincoords[s](t, 2), fincoords[s](t, 3))>[3]+
      pfincoords[s-1](nb, 3)>
    end do
```

```
        end if;
        pfincoords := [op(pfincoords), pfin]
      end do;
> for t to nbinMb-1 do
    print(pfincoords[t][nbinkb+1, 1]);
    print(pfincoords[t+1][1, 1])
  end do
> finalpt := Matrix(nbinkb*nbinMb, 3);
  finalv := Vector(nbinkb*nbinMb);
> for t to nbinMb do
    for l to nbinkb do
      finalpt(l+nbinkb*(t-1), 1) := pfincoords[t][l, 1];
      finalpt(l+nbinkb*(t-1), 2) := pfincoords[t][l, 2];
      finalpt(l+nbinkb*(t-1), 3) := pfincoords[t][l, 3];
      finalv(l+nbinkb*(t-1)) := <finalpt(l+nbinkb*(t-1), 1),
      finalpt(l+nbinkb*(t-1), 2), finalpt(l+nbinkb*(t-1), 3)>
    end do
  end do;
> finalcoordinates := Matrix(100, 3);
> for l to 100 do
    finalcoordinates(l, 1) := finalv(l)[1];
    finalcoordinates(l, 2) := finalv(l)[2];
    finalcoordinates(l, 3) := finalv(l)[3]
  end do;
> ExportMatrix("finalmatrix.txt", finalcoordinates, target = delimited);
> for l from 2 to nbinkb*nbinMb do
    normv(l):=norm(<finalv(l)[1]-finalv(l-1)[1],finalv(l)[2]-finalv(l-1)[2],
    finalv(l)[3]-finalv(l-1)[3]>,2):
    print(normv(l),l-1):
  end do:
> pointplot3d([finalv(1), finalv(2), finalv(3), finalv(4),  finalv(5),
finalv(6), finalv(7), finalv(8), finalv(9), finalv(10), finalv(11),
finalv(12), finalv(13), finalv(14), finalv(15), finalv(16), finalv(17),
finalv(18), finalv(19), finalv(20), finalv(21), finalv(22), finalv(23),
finalv(24), finalv(25), finalv(26), finalv(27), finalv(28), finalv(29),
finalv(30), finalv(31), finalv(32), finalv(33), finalv(34), finalv(35),
finalv(36), finalv(37), finalv(38), finalv(39), finalv(40), finalv(41),
finalv(42), finalv(43), finalv(44), finalv(45), finalv(46), finalv(47),
finalv(48), finalv(49), finalv(50), finalv(51), finalv(52), finalv(53),
finalv(54), finalv(55), finalv(56), finalv(57), finalv(58), finalv(59),
finalv(60), finalv(61), finalv(62), finalv(63), finalv(64), finalv(65),
finalv(66), finalv(67), finalv(68), finalv(69), finalv(70), finalv(71),
finalv(72), finalv(73), finalv(74), finalv(75), finalv(76), finalv(77),
finalv(78), finalv(79), finalv(80), finalv(81), finalv(82), finalv(83),
finalv(84), finalv(85), finalv(86), finalv(87), finalv(88), finalv(89),
finalv(90), finalv(91), finalv(92), finalv(93), finalv(94), finalv(95),
finalv(96), finalv(97), finalv(98), finalv(99), finalv(100)],
color = [blue], scaling = constrained, axes = normal);
```